I CLAIM:

1. A method for assessing a relationship of an interface to a unit, wherein a software program comprises plural units, and wherein a unit exposes one or more interfaces, the method comprising:

detecting a reference to an interface of a unit of software;

determining if the unit that exposes the interface is known by checking a data structure comprising one or more entries, wherein an entry maps an interface to a unit identity;

if the unit that exposes the interface is not known,

discovering the unit identity of the unit that exposes the interface;

adding an entry to the data structure, wherein the entry maps the interface to the discovered unit identity; and

performing an operation based upon an entry of the data structure.

2. A computer-readable medium having computer-executable instructions for performing the method of claim 1.

3. The method of claim 1 wherein the step of detecting comprises:

noting one or more return parameters from a called function; and

parsing the one or more return parameters to detect a reference to an interface.

4. The method of claim 3 wherein the called function is a unit creation function.

5. The method of claim 1 wherein the step of detecting comprises:

noting one or more outgoing parameters to a called function; and

parsing the one or more outgoing parameters to detect a reference to an interface.

6. The method of claim 1 wherein the data structure is a hash table, and wherein the step of determining comprises:

hashing the detected reference;

if the detected reference hashes to a unit identity, returning a value that indicates the unit identity of the unit that exposes the interface is known; and

if the detected reference does not hash to a unit identity, returning a value that indicates the unit identity of the unit that exposes the interface is not known.

7. The method of claim 1 wherein the data structure is a hash table, wherein the step of adding an entry comprises:

creating a new entry in the hash table, wherein the new entry associates the interface with the discovered unit identity.

8. The method of claim 1 wherein an interface wrapper stores data comprising a unit identity, wherein the data structure is a hash table for associating interfaces with interface wrappers, and wherein the step of determining comprises:

hashing the detected reference;

if the detected reference hashes to an interface wrapper, returning a reference to the interface wrapper; and

if the detected reference does not hash to an interface wrapper, returning a value indicating that the interface does not have an interface wrapper.

9. A computer-readable medium having computer-executable instructions for performing the method of claim 8.

10. The method of claim 1 wherein an interface wrapper stores data comprising a unit identity, wherein the data structure is a hash table for associating interfaces with interface wrappers, the method further comprising:

if the unit that exposes the interface is not known,

creating an interface wrapper, wherein the interface wrapper stores data comprising the discovered unit identity; and

during the step of adding an entry, creating a new entry in the hash table, wherein the new entry associates the interface with the created interface wrapper.

11. The method of claim 1 wherein a local variable stores data comprising the unit identity of the unit from which the detected reference originated, and wherein the step of discovering the unit identity comprises noting the value stored in the local variable.

12. The method of claim 11 wherein an instrumentation system provides the unit identity of the unit from which the detected reference originated.

13. The method of claim 1 further comprising:

if the unit that exposes the interface is known,

verifying the unit identity of the unit that exposes the interface; and

overwriting an entry of the data structure, wherein the entry maps the interface to the verified unit identity.

14. The method of claim 1 wherein the step of performing an operation based upon an entry of the data structure comprises:

detecting a communication passing through the interface;

measuring the size of the communication;

determining the unit identity of the unit that exposes the interface from the entry of the data structure;

associating the measured size with the unit that exposes the interface.

15. The method of claim 1 wherein the step of performing an operation based upon an entry of the data structure comprises:

receiving a call to a unit activation function from a client unit;

determining the unit identity of the client unit from the entry of the data structure;

classifying the unit to be activated based upon the unit identity of the client unit, resulting in a classifier;

determining a location in a distributed computing environment using the classifier; and

routing the call to the location.

16. A method for assessing a relationship of an interface to a unit, wherein a software program comprises plural units, and wherein a unit exposes one or more interfaces, the method comprising:

on receiving a reference to an interface as a return parameter from a function call,

determining if the unit that exposes the interface is known by checking a data structure comprising one or more entries, wherein an entry maps an interface to a unit identity;

if the unit that exposes the interface is not known,

discovering the unit identity of the unit that exposes the interface;

adding an entry to the data structure, wherein the entry maps the interface to the discovered unit identity; and

performing an operation based upon an entry of the data structure.

17. A computer-readable medium having computer-executable instructions for performing the method of claim 16.

18. The method of claim 16 wherein the data structure is a hash table, and wherein the step of determining comprises:

hashing the detected reference;

if the detected reference hashes to a unit identity, returning a value that indicates the unit identity of the unit that exposes the interface is known; and

if the detected reference does not hash to a unit identity, returning a value that indicates the unit identity of the unit that exposes the interface is not known.

19. The method of claim 16 wherein the data structure is a hash table, wherein the step of adding an entry comprises:

creating a new entry in the hash table, wherein the new entry associates the interface with the discovered unit identity.

20. A method for assessing a relationship of an interface to a unit of software using an interface wrapper, wherein the software comprises plural units, and wherein a unit of software exposes one or more interfaces, the method comprising:

detecting a reference to an interface of a unit of software;

determining if the interface is wrapped using a hash table for associating an interface with an interface wrapper;

if the interface is not wrapped,

creating an interface wrapper for the interface, wherein the interface wrapper stores a reference to instrumentation and the reference to the interface;

creating a new entry in the hash table, wherein the new entry associates the interface with the created interface wrapper;

discovering the unit identity of the unit that exposes the interface;

storing in the interface wrapper the unit identity of the unit that exposes the interface;

providing to a client unit a reference to the interface wrapper; and

receiving from the client unit an invocation of the instrumentation through the reference to the interface wrapper.

21. A computer-readable medium having computer-executable instructions for performing the method of claim 20.

22. The method of claim 20 wherein the step of determining whether the interface is wrapped comprises:

hashing the detected reference;

if the detected reference hashes to an interface wrapper, returning a reference to the interface wrapper; and

if the detected reference does not hash to an interface wrapper, returning a value that indicates the interface is not wrapped.

23. The method of claim 20 wherein the reference to instrumentation comprises a pointer to a table comprising at least one pointer to one or more instrumentation functions.

24. The method of claim 20 wherein the step of creating an interface wrapper further comprises storing in the interface wrapper a type description of the interface.

25. The method of claim 20 wherein a local variable stores the unit identity of the unit from which the detected reference originated, and wherein the step of discovering the identity of the unit that exposes the interface comprises noting the value stored in the local variable.

26. The method of claim 20 wherein the invocation comprises a communication from the client unit directed towards the unit that exposes the interface, the method further comprising:

measuring the size of the communication using the instrumentation;

associating the measured size with the unit that exposes the interface and the client unit; and

calling the unit that exposes the interface.

27. The method of claim 26 further comprising:

before the step of calling the unit, setting a return address for returning from the called unit as a reference to the instrumentation;

after the step of calling the unit, receiving from the unit that exposes the interface a second invocation of the instrumentation, wherein the second invocation comprises a second communication from the unit to the client unit;

measuring the size of the second communication using the instrumentation;

associating the measured size of the second communication with the unit that exposes the interface and the client unit; and

returning control to the client unit.

28. A computer-readable medium having computer-executable instructions for performing the method of claim 27.

29. A computer–readable medium having stored thereon a data structure, comprising:

a first data field containing data representing a reference to instrumentation;

a second data field containing data representing a reference to an interface of a unit of software; and

a third data field containing data representing an identity of the unit of software.

30. The computer-readable medium of claim 29 wherein the reference to instrumentation comprises a pointer to a table comprising at least one pointer to one or more instrumentation functions.

31. The computer-readable medium of claim 29 further comprising a fourth data field containing data representing a reference to a type description file for the interface.

32. The computer-readable medium of claim 31 wherein during a profiling operation on the software, the instrumentation references the type description file to parse and measure one or more parameters passed across the interface.

33. The computer-readable medium of claim 29 wherein during a profiling operation on the software, the instrumentation references the identity of the unit to associate a communication across the interface with the unit.

34. The computer-readable medium of claim 29 wherein during an activation operation for a new unit of the software, the instrumentation references the identity of the unit to classify the new unit.

35. The computer-readable medium of claim 29 further comprising a fourth data field containing data representing a log of activity over the interface.

36. A method of instrumenting one or more units of an application program, wherein a unit comprises one or more interfaces through which communications to and from the unit pass, and wherein a client unit requests creation of a server unit, the method comprising:

intercepting a call from a client unit to a unit creation function to create a server unit;

routing the call to the unit creation function, wherein the unit creation function creates the server unit and returns a reference to an interface of the server unit;

detecting the reference to the interface returned from the unit creation function;

creating an interface wrapper for the interface, the interface wrapper comprising a reference to instrumentation, the reference to the interface, and a reference to a type description of the interface; and

returning to the client unit a reference to the interface wrapper in place of the reference to the interface, wherein the client unit treats the reference to the interface wrapper as the reference to the interface.

37. A computer-readable medium having computer-executable instructions for performing the method of claim 36.

38. The method of claim 36 wherein an interface comprises at least one function, the method further comprising:

receiving from the client unit an invocation of the instrumentation that is based upon the reference that the client unit received, wherein the client unit treats the invocation as a call to a function of the interface, and wherein the client unit passes one or more parameters for the function; and

executing the instrumentation, wherein the instrumentation parses and measures the one or more parameters based upon the type description of the interface, producing a measurement.

39. The method of claim 38 further comprising:

calling the function of the interface using the reference to the interface in the interface wrapper, wherein the function performs an operation and returns one or more return parameters;

executing the instrumentation, wherein the instrumentation parses and measures the one or more return parameters based upon the type description of the interface, producing a second measurement; and

returning execution to the client unit.

40. A computer-readable medium having computer-executable instructions for performing the method of claim 39.

41. The method of claim 39 wherein the reference to instrumentation comprises a pointer to a table of at least one pointer to one or more instrumentation functions, and wherein the invocation of the instrumentation comprises a call to an instrumentation function.

42. The method of claim 39 further comprising:

before the step of executing the instrumentation, comparing the reference to the interface wrapper with the invocation of the instrumentation to calculate an offset;

determining from the offset the identity of the function of the interface to be called;

during the step of executing the instrumentation, parsing and measuring based upon type description for the function to be called; and

during the step of calling the function, applying the offset to the reference to the interface when referencing the function of the interface.

43. The method of claim 39 wherein the interface wrapper further comprises data identifying the server unit, the method further comprising:

associating the measurements with the server unit.

44. The method of claim 39 wherein the interface wrapper further comprises data identifying the server unit, and wherein a local variable stores an identifier for the client unit, the method further comprising:

associating the measurements with the client unit and the server unit.

45. The method of claim 44 wherein the local variable is a stack, and wherein an identifier for the client unit is at the top of the stack, the method further comprising:

before the step of calling the function, pushing an identifier for the server unit on the stack; and

after the step of calling the function, popping the identifier of the server unit off the stack.

46. A computer-readable medium having computer-executable instructions for performing the method of claim 45.

47. The method of claim 36 wherein a dynamic structure tracks the state of the application program, and wherein a local variable stores an identifier for the client unit, the method further comprising:

before the step of routing the call to the unit creation function,

dynamically classifying the server unit based upon the dynamic structure and the local variable, resulting in a unit classifier;

mapping the unit classifier to a location in a distributed computing environment; and

during the step of routing the call to the unit creation function, routing the call to the location.

48. The method of claim 47 wherein the dynamic structure is a call stack comprising one or more client unit invocation frames, and wherein the step of dynamically classifying the server unit comprises:

traversing the call stack;

noting a return address for an invocation frame;

examining the local variable;

noting the identifier for the client unit; and

creating a unit classifier based upon the return address and the identifier.

49. The method of claim 36 further comprising:

receiving from the client unit an invocation of the instrumentation that is based upon the reference that the client unit received, wherein the client unit treats the invocation as a call to a function of the interface;

determining if the interface is documented based upon the interface wrapper;

if the interface is undocumented,

transferring execution to the function of the interface using the reference to the interface in the interface wrapper, wherein the function performs an operation and finishes; and

returning execution to the client unit.

50. The method of claim 36 wherein the interface comprises one or more functions, wherein a call stack holds one or more parameters of a call to a function from a client unit, wherein the call stack holds a return address for the client unit, and wherein a stack pointer points to the top of the stack, the method further comprising:

receiving from the client unit an invocation of the instrumentation that is based upon the reference that the client unit received, wherein the client unit treats the invocation as a call to a function of the interface, wherein the client unit pushes

the one or more parameters for the call to the function on the call stack, and wherein the client unit pushes a return address for the client unit on the call stack;

determining if the interface is documented based upon the interface wrapper;

if the interface is undocumented,

storing the value of the stack pointer;

storing the return address for the client unit;

replacing the return address for the client unit on the call stack with a return address for an instrumentation function;

transferring execution to the function of the interface using the reference to the interface in the interface wrapper, wherein the function performs an operation;

after the function finishes,

popping the return address for the instrumentation function from the call stack;

popping the one or more parameters from the call stack;

transferring execution to the instrumentation function;

calculating the size of the one or more parameters based upon the stored value of the stack pointer and the current value of the stack pointer;

storing the calculated size in the type description of the interface wrapper;

transferring execution to the client unit.

51. A computer-readable medium having computer-executable instructions for performing the method of claim 50.

52. The method of claim 50 wherein the step of determining if the interface is documented comprises:

examining the type description referenced in the interface wrapper; and

if the type description contains no type information, indicating that the interface is undocumented.

53. The method of claim 50 wherein the step of determining if the interface is documented comprises:

examining the type description referenced in the interface wrapper; and

if the type description contains opaque type information, indicating that the interface is undocumented.

54. The method of claim 50 wherein the interface wrapper further comprises an identifier for the server unit that exposes the interface, and wherein a local variable stores an identifier for the client unit, the method further comprising:

if the interface is undocumented,

noting the relationship of the undocumented interface to the client unit and the server unit based upon the identifiers; and

noting a pair-wise location constraint between the client unit and the server unit.

55. The method of claim 36 wherein the interface wrapper further comprises an identifier for the server unit that exposes the interface, and wherein a local variable stores an identifier for the client unit, the method further comprising:

receiving from the client unit an invocation of the instrumentation based upon the reference that the client unit received;

determining if the interface is documented based upon the interface wrapper;

if the interface is undocumented,

noting the relationship of the undocumented interface to the client unit and the server unit based upon the identifiers; and

noting a pair-wise location constraint between the client unit and the server unit.

56. A computer-readable medium having computer-executable instructions for performing the method of claim 55.

57. A method for handling an undocumented interface between a client unit and a server unit of an application program, wherein the interface comprises one or more functions, and wherein a description file comprises a description of the interface, the method comprising:

receiving from a client unit a call to a function of an interface;

determining if the interface is documented based upon the description of the description file;

if the interface is undocumented,

transferring execution to the function, wherein the function performs an operation; and

after the function completes,

transferring execution to the client unit.

58. A computer-readable medium having computer-executable instructions for performing the method of claim 57.

59. The method of claim 57 wherein a call stack holds one or more parameters of the call to the function pushed by the client unit, and wherein a stack pointer points to the top of the stack, the method further comprising:

if the interface is undocumented, before the step of transferring execution to the function, storing the value of the stack pointer;

after the function completes, before the step of transferring execution to the client unit,

popping the one or more parameters from the call stack;

calculating the size of the one or more parameters based upon the stored value of the stack pointer and the current value of the stack pointer; and

storing the calculated size in the description file.

60. The method of claim 57 wherein a call stack holds a return address for the client unit pushed by the client unit, the method further comprising:

if the interface is undocumented, before the step of transferring execution to the function,

storing the return address for the client unit;

replacing the return address for the client unit on the call stack with a return address for an instrumentation function;

after the function completes, before the step of transferring execution to the client unit,

popping the return address from the call stack; and

transferring execution to the instrumentation function.

61. The method of claim 57 wherein a call stack holds one or more parameters of the call to the function pushed by the client unit, wherein the call stack holds a return address for the client unit pushed by the client unit, and wherein a stack pointer points to the top of the stack, the method further comprising:

if the interface is undocumented, before the step of transferring execution to the function,

storing the value of the stack pointer;

storing the return address for the client unit;

replacing the return address for the client unit on the call stack with a return address for an instrumentation function;

after the function completes, before the step of transferring execution to the client unit,

popping the return address from the call stack; and

popping the one or more parameters from the call stack;

transferring execution to the instrumentation function;

calculating the size of the one or more parameters based upon the stored value of the stack pointer and the current value of the stack pointer; and

storing the calculated size in the description file.

62. A computer-readable medium having computer-executable instructions for performing the method of claim 61.

63. The method of claim 57 wherein the step of determining if the interface is documented comprises:

examining the description file; and

if the description file contains no description, indicating that the interface is undocumented.

64. The method of claim 57 wherein the step of determining if the interface is documented comprises:

examining the description file; and

if the description file contains inadequate information for parsing a communication across the interface, indicating that the interface is undocumented.

65. The method of claim 57 wherein a data structure stores an identifier for the client unit and an identifier for the server unit, the method further comprising:

if the interface is undocumented,

noting the relationship of the undocumented interface to the client unit and the server unit based upon the identifiers; and

noting a pair-wise location constraint between the client unit and the server unit.

66. A computer-readable medium having computer-executable instructions for performing the method of claim 65.